

APPENDIX A

BRAINS CONSTELLATION MODELER MANUAL

The main usage of *BRAINS* constellation modeler is to build a template model as well as a linear estimation model from training datasets for *BRAINS* constellation detector (in **Appendix B**). The tool belongs to the *BRAINSConstellationDetector* project [1] that is one of a series in the *BRAINS* tool suite registered at *NITRC* [2]. The models can be simply considered as some statistical information extracted from the training datasets. This appendix will discuss the typical usage of the tool by how to accomplish the following common tasks:

- How to generate a template model for customized landmarks
- How to generate a linear estimation model for customized landmarks

Users can skip this manual if they prefer to use default model files.

A.1 Generate a template model for customized landmarks

If users plan to create their own template model, first note that the tool to achieve this goal is command line-based, so the basic usage looks like this:

```
1 ${BINARY_DIR}/./BRAINSConstellationModeler \
  OPTION1 [ARGUMENT1] [OPTION2 [ARGUMENT2]] ...
```

Here, "[]" means that the component enclosed is optional. For more usage information, type the following command in a terminal:

```
${BINARY_DIR}/./BRAINSConstellationModeler -h
```

The main input for the modeler is a few *NIfTI-1* formatted [3] image files to be trained and some files containing the landmarks locations of the images. Users can input this important information by specifying the filename of a list that contains the

filename of each image and landmark file to be used for training. The corresponding option is *-i* or *--inputTrainingList*. Note the training list file should be in a simple document format such as *txt*. Users need to enter three categories of information in the file. They are

- General training parameters
- Template size
- Image and landmark filenames

The definition of the training list file is summarized as follows:

```

1 numberOfDatasets searchBoxDims resolutionUnits
  initialRotationAngle initialRotationStep numRotationSteps
3
  landmarkName_1 templateRadius templateHeight
5 landmarkName_2 templateRadius templateHeight
  ...
7 END

9 trainingImageFilename_1
  traininglandmarkFilename_1
11 END
  trainingImageFilename_2
13 traininglandmarkFilename_2
  END
15 ...

```

The landmark file is supported by *NA-MIC* [4], and can be created and modified by some popular medical image viewer with *Fiducials* module such as the *3DSlicer* software [5]. An example of how it may look like is shown in the following listing:

```

1 # Fiducial List file /ACPCModel/data/test.fcsv
  # version = 2
3 # name = test.fcsv
  # numPoints = 19
5 # symbolScale = 5
  # symbolType = 13
7 # visibility = 1
  # textScale = 4.5
9 # color = 0.4,1,1
  # selectedColor = 1,0.5,0.5
11 # opacity = 1

```

```

# ambient = 0
13 # diffuse = 1
# specular = 0
15 # power = 1
# locked = 0
17 # numberingScheme = 0
# columns = label , x , y , z , sel , vis
19 AC, -127.032, -128.216, 125.942, 1, 1
PC, -125.988, -155.846, 125.756, 1, 1
21 MPJ, -126.098, -152.606, 113.279, 1, 1
...

```

The other compulsory argument for generating a user-specified template model file is to indicate the desired output template model filename by using the `--outputModel` option.

A.2 Generate a linear estimation model for customized landmarks

The linear estimation model builds the linear morphometric relationship among landmarks by applying EPCA to their instances in training datasets. Along with the local search module, this model is used to estimate the search centers for newly introduced or customized landmarks iteratively. To build a linear estimation model for customized landmarks, user should modify two files, named *load_landmarks.m* and *train_LM_EPCA.m* in the training model source directory. To ensure a successful training, the modeler requires the accurate information of six mandatory landmarks. They are *ac*, *pc*, *MPJ*, *4VN*, *LE*, and *RE*.

The *load_landmarks.m* is a *Matlab* [6] script file that tells the modeler which landmarks should be taken into account. An example of the script is shown in the following listing:

```

%% Initialization
2 AC = [];
PC = [];
4 ...
% Add any customized landmarks here
6
%% Add landmarks location info from training datasets

```

```

8 % This following line will try to add a new
  % AC landmark (-127.164,-127.69,127.285)
10 AC = [AC; -127.164,-127.69,127.285];
  AC = [AC; -127.032,-128.216,125.942];
12 ...
  PC = [PC; -126.642,-155.162,126.514];
14 PC = [PC; -125.988,-155.846,125.756];
  ...
16 % Add any customized landmarks location info here

18 %% Transform landmarks from RAS space to LPS space
  %if the landmark location is obtained by 3DSlicer
20 AC(:,1) = -AC(:,1);
  AC(:,2) = -AC(:,2);
22 PC(:,1) = -PC(:,1);
  PC(:,2) = -PC(:,2);
24 ...
  % Transform any customized landmarks here

```

The *train_LM_EPCA.m* script is used to compute the principal components of the landmark vector space and, find the optimal linear model parameters in terms of least squares in each iteration.

To train the linear model using EPCA, in *Matlab* directly run script in terminal:

```

1 train_LM_EPCA
  LME_EPCA
3 save('LME_EPCA.mat', '*','-v4')

```

The scripts will generate two model files:

processingList.txt contains light-weighted information like the name of the EPCA landmarks and the search radii of them

LME_EPCA.mat contains matrix and vector variables that can be load by vxl matlab I/O in the detector

The model files in the source directory will be copied to the binary directory each time the build of detector is updated.

The content of *processingList.txt* file looks as follows,

```

1 aq_4V
  1.60000000000000000008881784197001252
3 genu
  4.47740183533491542533511164947413
5 rostrum
  3.09800088204989743090322917851154
7 BPons
  2.97690634742662885159347752050962
9 optic_chiasm
  3.64155342814196103518042946234345
11 ...

```

The content of *LME_EPCA.mat* file looks as follows,

	Name	Value
1	aq_4V__s	[−0.2394, −28.6931, −5.1403]
3	aq_4V__M	<3x15 double>
	genu__s	[−0.2036, −21.1673, −4.1565]
5	genu__M	<3x18 double>
	rostrum__s	[−0.2302, −24.2579, −0.7434]
7	rostrum__M	<3x21 double>
	...	

APPENDIX B

BRAINS CONSTELLATION DETECTOR MANUAL

The *BRAINS* constellation detector has two identities: both as an image registration tool and as a constellation detector for human brain MR scans. The tool belongs to the *BRAINSConstellationDetector* project [1] that is one of a series in the *BRAINS* tool suite registered at *NITRC* [2]. This appendix will discuss the typical usage of the tool by how to accomplish the following common tasks:

- How to generate a *NIfTI-1*-formatted [3] acpc-aligned image
- How to generate an *ITK*-recognized [7] registration transformation
- How to generate a *NA-MIC*-supported [4] landmark file
- How to tune the parameters of the detector
- How to handle the failure of automated detection

The tool is a command line-based application, so the basic usage looks like this:

```

2  ${BINARY_DIR}/./BRAINSConstellationDetector \
   OPTION1 [ARGUMENT1] [OPTION2 [ARGUMENT2]] ...

```

Here, ‘‘[]’’ means that the component enclosed is optional. For more usage information, type the following command in a terminal:

```

${BINARY_DIR}/./BRAINSConstellationDetector -h

```

Some practical use cases are also provided in a tutorial webpage [8] that is hosted by *NITRC*. Users might want to look at them before dealing with their own tasks.

B.1 Generate a NIfTI-1-formatted acpc-aligned image

The detector works with *NIfTI-1* data format that is proposed by Neuroimaging Informatics Technology Initiative (NIfTI) [3]. Providing the filename of a *NIfTI-1*-formatted brain image with arbitrary image direction, spacing, and origin by using the `-i` or `--inputVolume` option, the detector is able to generate an acpc-aligned image automatically. Users can use the `--outputResampledVolume` option to obtain an isotropic and resampled aligned image, or they can use the `-o` or `--outputVolume` option to obtain an unresampled but direction cosine-modified image that has identical voxel contents as the input image has, and the same physical representation as the resampled image in the LPS coordinate system (*i.e.* in the physical space). Consistently choosing to apply the latter option can effectively reduce the resampling errors throughout a large image-processing pipeline. Its idea has already been adapted to many other *BRAINS* software. In addition, due to the large volume of a *3D* medical image, the resulted output image is compressed, usually featured by a *.nii.gz* filename extension.

B.2 Generate an ITK-recognized registration transformation

Users can obtain the resulted versor transformation from the output aligned image (*i.e.* the moving image) to the original input image (*i.e.* the fixed image) by using the `-t` or `--outputTransform` option. The resulted registration transformation will be written to a file with a *.mat* filename extension that is recognized by *ITK* [7] as well as *3DSlicer* [5].

B.3 Generate a NA-MIC-supported landmark list file

The resulted landmark list file is supported by National Alliance for Medical Image Computing (NA-MIC) [4] and is readable from the *Fiducials* module in its *3DSlicer* software [5]. The landmark list file is identified by a *.fcsv* filename extension. Users can use the `--outputLandmarksInInputSpace` option to generate a landmark list file in the input space where the input image lies. Alternatively, they can specify the output landmark list file for the *acpc*-aligned image by using the `--outputLandmarksInACPCAlignedSpace` option as long as the aligned image is requested as well.

B.4 Tune the parameters of the detector

The default parameters of the program are stored in an XML file along with the source files. Of course, a preset parameter set won't fit for all situations. In response to the problem, users are provided with the convenience to modify various parameters of the tool with proper options and arguments in command line. For instance, users can:

Specify a different template model file: We provide three kinds of built-in models: T_1 -weighted, T_2 -weighted, and PD. They are files ending with a *.mdl* filename extension, and can be found in `${TestData_DIR}`. The default model is set to the T_1 -weighted model. Users are free to switch from different kinds of models, or even use their own. To specify a different model file, users can use the `--inputTemplateModel` option. Also note that if the target input is a T_2 -weighted or PD dataset, users are responsible to set the Hough eye detector mode to 0 accordingly by specifying the option and argument like this:
`--houghEyeDetectorMode 0;`

Specify a different linear estimation model file: Users can specify a different

linear estimation model by using the `--inputEPCAModelTxt` and `--inputEPCAModelMat` options. The linear estimation model files are stored in the `{TestData_DIR}` directory, and named as *processingList.txt* and *LME_EPCA.mat*. The *txt* file includes light-weighted information as landmark names and their search radii; the *mat* file includes matrices and vectors information of the model. The definition of the model content is described in **Appendix A**.

Apply different interpolation methods: Users can apply different interpolation methods for the application. The available options are *Linear*, *NearestNeighbor*, *BSpline*, or *WindowedSinc*. The *Linear* interpolation method is set as default. Users can use the command option `--interpolationMode` to change the interpolation methods.

B.5 Handle the failure of automated detection

The constellation detector is designed in a way so that it can take the users' feedback as a hint for its own processing. Users can interact with the detector by applying related options and providing corresponding arguments. For instance, users can:

Provide user-specified landmarks information: When the detector is about to estimate a certain landmark, it will first try to fetch the corresponding information from users. It will simply skip the estimation phase for that landmark and use the right information provided by the user directly as long as it finds one in the user-specified landmark list file. Users can specify the landmark list file by using the option `--inputLandmarksEMSP`. Note the landmarks should be in the *EMSP*-aligned space. The detector requires this because many landmarks especially those who are midline points are much easier to pick, and the

resulted landmarks are considered to be more accurate in the *EMSP*-aligned space. Users can request the detector to initial a pair of landmark list file and a resampled input image in the *EMSP*-aligned space for later manual correction use by setting the “*write debugging images level*” option argument be greater than one. The resulted landmark list file is named as *EMSP.fcsv* and the resampled image is named as *EMSP.nii.gz*. Both of them can be found in the $\{\text{DEBUG_IMAGE_DIR}\}$ directory that can be set through proper option and argument. After correcting the landmark locations with some handy GUI tools such as the *BRAINS* Constellation corrector that will be discussed in **Appendix C**, users can feed the corrected *EMSP.fcsv* landmark list file to the detector by the `--inputLandmarksEMSP` option.

Set write debugging image level and debugging image directory: The “*write debugging image level*” option controls the amount of exposure for debugging information. The default level is set to 0 that means no debugging images would be written to disk. Users can specify the debugging image level by using the option `--writedebuggingImagesLevel`. Similarly, users can use the option `--resultsDir` to specify the path where the debugging images should be written. The two options will be very handy when the user wants to correct some estimation result or interact with the detector.

Notify the detector about the failure of eye detection: Although the eye detector has some simple and innate mechanism to check if the detected eye centers are reasonable, sometimes it still may fail. In this case users can notify the failure of the eye detection by specifying the option `--forceHoughEyeDetectorReportFailure`. This will force the detector to terminate the program after it has written to disk a pair of empty landmark list file and a resampled input image in the *EMSP*-aligned space for later manual

correction. The resulted landmark list file is named as *EMSP.fcsv* and the re-sampled image is named as *EMSP.nii.gz*. Both of them can be found in the `{DEBUG_IMAGE_DIR}` directory. User can load the two files and manually specify the correct eye center locations by the proposed GUI tool that is introduced in **Appendix C**. Finally, user may want to use the technique that is described in **Section B.5** to use the corrected eye centers information in the detection process.

APPENDIX C

BRAINS CONSTELLATION GUI CORRECTOR MANUAL

The *BRAINS* constellation GUI corrector is a lite medical image viewer that provides the convenience for the users to correct landmark locations in a visual and efficient way. The tool was developed within the *Qt* framework [9] and the slice viewers are rendered with the help of the *VTK* [10] APIs. It belongs to the *BRAIN-SConstellationDetector* project [1] that is one of a series in the *BRAINS* tool suite registered at *NITRC* [2]. This appendix will discuss the typical usage of the tool by how to accomplish the following common tasks:

- How to load a *NIfTI-1*-formatted [3] image and a *NA-MIC*-supported [4] landmark list file
- How to generate a corrected landmark list file
- How to interact with the GUI tool

The GUI tool is invoked from a terminal, so the basic usage looks like this:

```
1 ${BINARY_DIR}/./BRAINSConstellationDetectorGUI \
  OPTION1 [ARGUMENT1] [OPTION2 [ARGUMENT2]] ...
```

Here, "[]" means that the component enclosed is optional. For more usage information, type the following command in a terminal:

```
${BINARY_DIR}/./BRAINSConstellationDetectorGUI -h
```

Some practical use cases are also provided in a tutorial page [8] hosted by *NITRC*. Users might want to look at them before dealing with their own tasks.

C.1 Load a NIfTI-1-formatted image and a NAMI-supported landmark list file

Users can specify the filename of an input *NIfTI-1*-formatted brain image by using the *-i* or *--inputVolume* option. Similarly, they can use the *-l* or *--inputLandmarks* option to tell the corrector that landmark list file is associated with the input image. Note the input image and input landmarks in the landmark list file should all be in the *EMSP*-aligned space because many landmarks especially those who are midline points are much easier to pick, and the resulted landmarks are considered to be more accurate in the *EMSP*-aligned space. The constellation detector has an option to generate those files automatically for users. For details on how to generate these two files, please look at **Appendix B.5**.

C.2 Generate a corrected landmark list file

This task can be achieved either by specifying the filename of the list file in command line with an option of *-o* or *--outputLandmarks*, or users can take the advantage of the **Save** button in the GUI tool. For example, if the output landmark list filename is specified in advance, clicking on the **Save** button will save the landmark information to the file with the specified filename. Else, if the input landmark list filename is specified, clicking on the **Save** button will result in saving the changes to the original landmark list file. Else, clicking on the **Save** button will trigger the same dialog as clicking on the **Save As** button. Detailed information on how to interact with the GUI tool will be discussed right away in the next section.

C.3 Interact with the GUI tool

The GUI tool is mainly composed of an image volume information module, a landmark list module, and a slice viewer module containing three viewers with axial, sagittal, and coronal views respectively. The modules are arranged as follows:

- The image information is shown at the left upper side;
- The landmarks list is shown at the left lower side;
- Three different viewers are located at the right upper side;
- The slider bars for each viewer are just below the viewers;
- All the buttons are at the bottom of the window.

A screenshot of the corrector is shown in **Fig. C.1**. Users can perform the following typical operations on landmark(s):

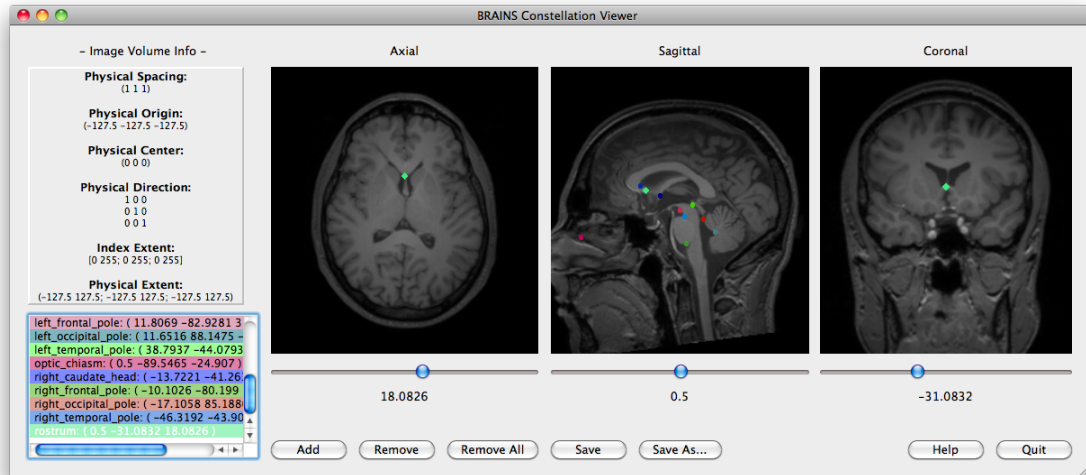


Figure C.1: A screenshot for *BRAINS* constellation GUI corrector

Adding a new landmark: Click the **Add** button at the bottom of the tool.

Selecting a landmark: Click on the landmark in the label list.

Moving a landmark: Select the landmark, move either one of the slider bar to find a proper slice, then click at a location in the corresponding viewer to move the landmark to the current place. Adjust the 3D location of the landmark with the help of different viewers.

Zooming in/out a viewer: Hover the mouse on the viewer, then right click then drag up/down to zoom in/out about the center.

Removing a landmark: Double click on the landmark in the label list, or select the landmark, then click on the **Remove** button at the bottom of the GUI. Click the **Accept** button when a follow-up dialog appears.

Others: Other operations such as **Remove All** landmarks, **Save** landmarks, etc can be achieved by clicking proper buttons at the bottom of the GUI. These basic operations can also be viewed inside the GUI corrector by clicking on the **Help** button.

REFERENCES

- [1] Nitrc wiki page for brains constellation detector. <http://www.nitrc.org/plugins/mwiki/index.php?title=brainscdetector:MainPage>.
- [2] Homepage for brains software. <http://www.nitrc.org/projects/brains/>.
- [3] Homepage for nifti, the neuroimaging informatics technology initiative. <http://nifti.nimh.nih.gov/>.
- [4] Homepage for na-mic, the national alliance for medical image computing. <http://www.na-mic.org/>.
- [5] Homepage for 3d slicer. <http://www.slicer.org/>.
- [6] Homepage for matlab. <http://www.mathworks.com/>.
- [7] Homepage for insight segmentation and registration toolkit (itk). <http://www.itk.org/Insight/>.
- [8] Tutorial page for brains constellation detector. <http://www.nitrc.org/plugins/mwiki/index.php/brainscdetector:Tutorial>.
- [9] Homepage for qt. <http://qt.nokia.com/>.
- [10] Homepage for visualization toolkit (vtk). <http://www.vtk.org/>.